IN THE CLAIMS

Please amend the claims as follows:

1. (previously presented) A method for automatically altering execution flow of a first computer program while the first program is executing, the method comprising the steps of:

executing the first program as a child of a second program, wherein instructions for the executing first program reside in respective memory locations of a computer and correspond to respective source code locations of the first program, and wherein the second program is operable for altering memory locations of the computer allocated to the child program, including memory designated for registers of the child program and memory designated for instructions of the child program;

receiving by the second program a desired flow of execution for the first program from a user predefined list, wherein the list includes a plurality of jump instructions, each jump instruction defining respective originating and destination source code locations in the first program; and

inserting the plurality of jump instructions into memory locations of the computer corresponding to respective originating source code locations of the first program defined by the respective jump instructions, wherein for each of the plurality of inserted jump instructions, execution flow in the first program jumps from the memory location corresponding to the respective jump instruction's originating source code location to the memory location corresponding to the respective jump instruction's defined destination source code location responsive to the first program encountering the memory location corresponding to the respective jump instruction's originating source code location, wherein the inserting is performed automatically by the second program responsive to the user predefined list.

2. (previously presented) The method as claimed in claim 1, wherein each jump instruction also defines a location for temporary storage of one or more instructions corresponding to the respective jump instruction's originating source code location of the first program, the method further comprising the step of temporarily storing, for each of the plurality of inserted jump instructions, an original instruction of the first program corresponding to the respective jump instruction's originating source code location.

3. (canceled)

4. (previously presented) The method as claimed in claim 2, further comprising the step of restoring the temporarily stored original instruction at the origin address after the step of inserting .

5-6. (canceled)

7. (previously presented) The method as claimed in claim 1, wherein for each of the plurality of inserted jump instructions, the method further comprises the step of adding the memory location corresponding to the respective jump instruction's originating source code location to a debugging register.

8. (previously presented) The method as claimed in claim 7, wherein for each of the plurality of inserted jump instructions, the method further comprises the step of removing the memory location corresponding to the respective jump instruction's originating source code location from a debugging register.

9. (previously presented) The method as claimed in claim 1, wherein for each of the plurality of inserted jump instructions, the method further comprises the step of clearing the jump instruction from the computer program.

10-11. (canceled)

12. (previously presented) The method as claimed in claim 1, further comprising the step of loading the first computer program as a child process of the second program.

13-27. (canceled)